**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра автоматики та управління в технічних системах**

«До захисту допущено»

Завідувач кафедри

_____ Олександр РОЛІК

«___» _____20__ р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Програмне забезпечення інформаційно-комунікаційних систем»**
**спеціальності 121 «Інженерія програмного забезпечення»**
**на тему: «Застосунок конвертації та програвання аудіо-файлів»**

Виконав:
студент IV курсу, групи ІТ-61
Яценко Іван Андрійович _____

Керівник:
Доц. каф. АУТС, к.т.н., доц.,
Савчук Олена Володимирівна _____

Рецензент:
Проф. каф. ОТ, д.т.н., проф.,
Сімоненко Валерій Павлович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Автоматики та управління в технічних системах**

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки – 121 « Програмна інженерія»

Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

**ЗАВДАННЯ**

**на дипломний проєкт студенту**

**Яценку Івану Андрійович**

1. Тема проєкту «Застосунок конвертації та програвання аудіофайлів», керівник проєкту доцент Савчук Олена Володимирівна, затверджені наказом по університету від «_07_»_травня____ 2020_ р. №__1081-с___

2. Термін подання студентом проєкту 9 червня 2020 року_____

3. Вихідні дані до проєкту

Програмне забезпечення на клієнтській архітектурі. Мова програмування C#, середовище розробки Visual Studio, обрані технології – .NET, Windows Presentation Foundation, Windows Media Foundation, бібліотека MF.NET.

4. Зміст пояснювальної записки

1. Вступ 2.Огляд предметної області 3. Огляд існуючих рішень 4. Аналіз вимог до програмного забезпечення 5. Проєктування застосунку 6. Вибір технологій розробки 7. Реалізація програмного забезпечення 8. Використання застосунку 9. Висновок

5. Перелік графічного матеріалу

Діаграма використання, діаграма класів, діграма послідовності конвертації аудіофайлу, діаграма послідовності редагування аудіофайлу, діаграма діяльності

6. Дата видачі завдання _____

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1 | Вибір тематичного напрямку та узгодження теми дипломного проекту | 10.03.2020 | |
| 2 | Аналіз теоретичних матеріалів та вивчення предметної області | 20.03.2020 | |
| 3 | Розробка технічного завдання, вибір методів та засобів реалізації задачі | 10.04.2020 | |
| 4 | Огляд існуючих рішень з тематики роботи | 26.04.2020 | |
| 5 | Розробка структури прототипу та проектування системи | 03.05.2020 | |
| 6 | Реалізація проекту | 10.05.2020 | |
| 7 | Налагодження та перевірка програми | 14.05.2020 | |
| 8 | Оформлення пояснювальної записки | 24.05.2020 | |
| 9 | Передзахист дипломного проекту | 09.06.2020 | |

Студент Іван ЯЦЕНКО

Керівник проєкту Олена САВЧУК

# АНОТАЦІЯ

Яценко І.А. Застосунок конвертації та програвання аудіофайлів. КПІ ім. Ігоря Сікорського, Київ, 2020.

Ключові слова: клієнтський застосунок, конвертація аудіофайлів, кодек, .Net, Windows Media Foundation, Windows Presentation Foundation.

Основна частина документу викладена у пояснювальній записці, що містить 61 сторінку та 25 таблиць.

Наведено огляд існуючих рішень, їх детальний аналіз та порівняння. Виконано огляд технологій, на базі яких можливе створення застосунку для конвертації та програвання аудіофайлів. Застосунок було розроблено із використанням мови програмування C# та технологій .Net, Windows Media Foundation, Windows Presentation Foundation, бібліотеки MF.NET.

Готовий додаток відповідає поставленим до нього при розробці вимогам та містить увесь необхідний функціонал.

# ABSTRACT

Yatsenko I.A. “Audio transcoder and player application”. Igor Sikorsky KPI, Kyiv, 2020.

Keywords: client-side application, audio transcoding, codec, .Net, Windows Media Foundation, Windows Presentation Foundation.

The bulk of the document is presented in an explanatory note, which contains 61 pages and 25 tables.

An overview of existing solutions, their detailed analysis and comparison is given. An overview of technologies was done, based on which it is possible to develop an audio conversion application. An application was created using C# programming language on .Net

platform, using Windows Media Foundation and Windows Presentation Foundation technologies and MF.Net library.

Completed application meets the requirements, which were set for it during development, and contains all the necessary functions.

| Номер рядка | Фор. | Позначення | Найменування | Кіл. | № | Приміт ка |
|---|---|---|---|---|---|---|
| 1 | | | Документація загальна | | | |
| 2 | | | | | | |
| 3 | | | Знову розроблена | | | |
| 4 | | | | | | |
| 5 | А4 | ІТ61.310БАК.004 ПЗ | Застосунок конвертації та | 60 | | |
| 6 | | | програвання аудіо-файлів. | | | |
| 7 | | | Пояснювальна записка | | | |
| 8 | А3 | ІТ61.310БАК.004 Д1 | Застосунок конвертації та | 1 | | |
| 9 | | | програвання аудіо-файлів. Діаграма | | | |
| 10 | | | використання | | | |
| 11 | А3 | ІТ61.310БАК.004 Д2 | Застосунок конвертації та | 1 | | |
| 12 | | | програвання аудіо-файлів. Діаграма | | | |
| 13 | | | послідовності редагування аудіо | | | |
| 14 | А3 | ІТ61.310БАК.004 Д3 | Застосунок конвертації та | 1 | | |
| 15 | | | програвання аудіо-файлів. Діаграма | | | |
| 16 | | | послідовності конвертації аудіо | | | |
| 17 | А3 | ІТ61.310БАК.004 Д4 | Застосунок конвертації та | 1 | | |
| 18 | | | програвання аудіо-файлів. Діаграма | | | |
| 19 | | | класів | | | |
| 20 | А3 | ІТ61.310БАК.004 Д5 | Застосунок конвертації та | 1 | | |
| 21 | | | програвання аудіо-файлів. Діаграма | | | |
| 22 | | | активності | | | |
| 23 | | | | | | |
| 24 | | | | | | |

| | | | | | ІТ61.310БАК.004 ТП | | |
|---|---|---|---|---|---|---|---|
| *Зм.* | *Арк.* | *Прізвище* | *Підпис* | *Дата* | | | |
| *Розроб.* | | *Яценко І.А* | | | *Застосунок конвертації та програвання аудіо-файлів. Відомість технічного проєкту* | *Літ.* | *Лист* 1 / *Листів* X |
| *Перевірив.* | | *Савчук О.В.* | | | | | |
| *Н. кон.* | | | | | | *КПІ ім. Ігоря Сікорського кафедра АУТС гр. ІТ-61* | |
| *Затв.* | | | | | | | |

# Пояснювальна записка

## до дипломного проєкту

# на тему: «Застосунок конвертації та програвання аудіо-файлів»

Київ — 2020 року

**ЗМІСТ**

# ВСТУП

Аудіоінформація супроводжує людину впродовж усього її життєвого шляху — від самих пелюшок і до старості. Згідно із досліженнями 2019 року, що опубліковані у журналі «Applied Cognitive Psychology», правильно підібрана фонова музика сприяє креативному мишленню та здатна покращувати працездатність людей. Це підкреслює актуальність обраної теми, адже у сучасних умовах праці, коли все більше компаній створюють відкриті простори для колективної роботи, необхідно якісно усунути галас для забезпечення сталого робочого процесу.

Аудіо записи широко застосовуються у індустрії розваг та маркетингу: від створення багатоканальних звукових доріжок для сучасних кінотеатрів та обробки адуіо для кліпів у мережі Інтернет до оголошень, що передаються гучномовцями. Для кожної цілі використовуються окремі інструменти і деякі з них є заскладними для простих користувачів.

Із поширенням швидкісного доступу до Інтернету велику росповсюдженість набули сервіси для потокового прослуховування музики, аудіокниг. Вони використовують новітні кодеки для стиснення звукових даних. Завдяки цьому розмір файлів зменшується, забезпечується безперервна передача інформації, але погіршується якість звуку. Існують також формати, що кодують звук без втрат якості, але розмір файлів збільшується у рази. Тому актуальною є проблема розробки нових форматів кодування звуку, що будуть ефективно стискати дані без суттєвих втрат якості. Для подальшої їх розробки, необхідно зрозуміти як представляються звукові дані у комп’ютері та навчитися взаємодіяти з їх структурою.

Сучасні проблеми потребують сучасних рішень. І їх можна реалізувати за допомогою використання новітніх технологій. Конвертація аудіоінформації є складним завданням, для вирішення якого необхідно не тільки розробити алгоритми кодування та декодування даних, але й надати достатню потужність комп’ютера для забезпечення швидкодії процесу. Від вибору кодеків залежить яких змін буде зазнавати

аудіоінформація на фундаментальному рівні та, у наслідок, наскільки спотвореною буде хвиля звуку кінцевого формату файлу.

Саме тому темою дипломного проєкту було обрано створення застосунку, що надає можливість конвертувати аудіо файли та програвати їх. Цей застосунок також полегшує життя користувачів завдяки додатковому функціоналу, оскільки аудіоінформація представляється наочно, у вигляді хвилі звуку. Окрім цього, під час використання застосунку, окрім конвертації та програвання аудіофайлів, користувачі мають змогу проводити базові операції з редагування аудіо: вони можуть видалити необхідний фрагмент, або, навпаки, вставити цей фрагмент до файлу ще раз.

# 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

Звук — це фізичні коливання, тобто хвиля, що розповсюджується через повітря. Вона характеризується амплітудою та частотою. Амплітуда визначає, наскільки гучним буде звук, а частота хвилі визначає, наскільки високим буде звук. Людське вухо може розпізнати хвилі частотою від 30 Гц до 22000 Гц. Для перетворення звуку із аналогового формату хвилі до цифрового, необхідно кодувати цю інформацію.

## 1.1 Дослідження форматів зберігання аудіоінформації

Аудіо інформація може зберігатися у різних форматах файлів та з різним показником стиснення, і конвертація аудіо між форматами є дійсно складним завданням. Існують три основні групи форматів аудіофайлів:

- нестиснені формати — такі як WAV;
- формати із стисненням без втрат якості — FLAC;
- формати із стисненням з втратами якості — MP3;

### 1.1.1 Waveform Audio Format

Waveform Audio Format (WAVE, WAV) — найбільш розповсюджений звуковий формат. Використовується в ОС Windows для зберігання звукових файлів. Майже будь-яка сучасна програма, що працює зі звуком, може прочитати або записати формат WAV, тому цей формат дуже цікавий для розробників програмного забезпечення. Для записування звуку використовуються різні способи стиснення, оскільки звукові файли мають великий об’єм. Найпростіший спосіб стиснення - імпульсно-кодова модуляція (Pulse Code Modulation, PCM), але він не забезпечує достатньо гарного рівня стиснення.

### 1.1.2 MPEG-1/2 Layer-3

У 2000-і роки, людство могло спостерігати широке розповсюдження .mp3-файлів, як із легальних, так і з нелегальних джерел. Аудіоматеріали, закодовані у форматі MPEG-1/2 Layer-3, почали набувати поширення у світі з 1995 року, і вже у 1999 році термін «.mp3» був визнаний найбільш популярним пошуковим запитом. Про MP3 не раз згадували у багаточисленних статтях газет, періодичних видань та на телебаченні, у більшості через можливий руйнівний вплив цього формату на індустрію звукозапису. Згодом, mp3 став домінуючим форматом для розповсюдження аудіофайлів у мережі Інтернет.

MPEG, або Moving Pictures Experts Group — це група фахівців, щоб була створена ISO для розробки стандартів кодування аудіо та відео інформації. З 1988 року, MPEG займалася стандартизацією методик стиснення аудіо та відео. Стандарти кодування аудіо, розроблені цією групою знайшли застосування у багатьох сферах, таких як:

- трансляція цифрового аудіо;
- зберігання даних для трансляції;
- супровідне аудіо для цифрового телебачення;
- потокове аудіо для Інтернет;
- портативні аудіо-програвачі;
- зберігання музики у комп’ютерах та обмін нею.

Із усіх створених стандартів, найбільшого розповсюдження набули MPEG-1 та MPEG-2.

Створений у 1992 році, стандарт MPEG-1 описує загальну систему кодування аудіо, придатну для широкого застосування. У цьому стандарті визначено три рівні стиснення звуку, із яких третій рівень (MPEG-1 Layer-3) видає найбільш оптимізовані характеристики якості аудіо при відносно малому бітрейті (близько 128 кбіт/с для стерео-сигналу).

У 1994 році, MPEG-2 відзначив другу фазу розвитку стандарту і привніс багато нових концепцій кодування відео, наприклад, метод черезрядкової розгортки, що дозволило усунути надлишковість частоти переданих кадрів. Щодо аудіо, то MPEG-2 привніс 2 розширення до MPEG-1:

- зворотньо сумісне мультиканальне кодування додало можливість використання формату із мультиканальними конфігураціями, включаючи системи 5.1, що використовуються у кіноіндустрії;
- кодування за нижчих частот дискретизації розширило діапазон стандарту MPEG-1 трьома частотами 16 КГц, 22.05 КГц та 24 КГц, що покращило ефективність кодування за дуже низьких бітрейтів.

MP3 є форматом стиснення з втратами, тобто частина звукової інформації, яку вухо людини сприйняти не може або сприймається не всіма людьми, знищується.

Високий ступінь компактності MP3 в порівнянні з PCM 16 біт Стерео 44.1кГц (CD Audio) і йому подібними форматами при збереженні аналогічної якості звучання досягається за допомогою додаткового квантування за встановленою схемою, що дозволяє мінімізувати втрати якості. Ступінь стиснення, і, відповідно, обсяг додаткового квантування, визначаються не форматом, а користувачем при визначенні параметрів кодування.

MPEG-1 підтримує кодування аудіо при частотах дискретизації 32 КГц, 44.1 КГц та 48 КГц. MPEG-2 розширює цей діапазон половинними частотами, тобто 16 КГц, 22.05 КГц та 24 КГц.

MPEG-1/2 Layer-3 підтримує можливість стиснення аудіо як із сталим бітрейтом, так і змінним. Стандарт MPEG-1 визначає діапазон можливих бітрейтів від 32 кбіт/с до 320 кбіт/с, а стандарт MPEG-2 — від 8 кбіт/с до 160 кбіт/с.

MP3 підтримує двоканальне кодування, існують такі режими:

- моно - одноканальне кодування. При кодуванні двоканального матеріалу у такому режимі, усі відмінності між каналами будуть знищені;

– стерео - двоканальне кодування, кожен канал містить різну аудіо інформацію, бітрейт поділяється навпіл на кожен канал;

– об’єднане стерео - двоканальне кодування із поєднанням інформації двох каналів, що позитивно впливає на якість вихідного файлу у порівнянні із стерео режимом при однаковому бітрейті.

І хоча стиснення аудіо розглядають як основну технологію, що призвела до популярності формату, чималий внесок до цього також склало широке розповсюдження Інтернету, компакт-дисків, а також розвиток комп’ютерів, що спричинив можливість кодування програмними засобами. Основними причинами популярності формату можна виділити наступні:

– MPEG визначено як відкритий стандарт, що доступний для використання будь-ким, тому що жодна компанія не володіє стандартом;

– приклади вихідного коду було опубліковано у відкритий доступ, щоб допомогти розробникам уникнути неправильного розуміння тексту стандартів;

– за винятком деяких неповних розробок, не було отримано повідомлень від вендорів щодо проблем сумісності обладнання та програмного забезпечення;

– створення великої кількості апаратних, а згодом і програмних засобів кодування/декодування, що було обумовлено попитом з боку теле- та радіо-індустрії, які використовували формат для трансляцій.

Найбільша на сьогодні перевага MP3 перед іншими подібними форматами полягає в тому, що ні про один інший формат не можна поки впевнено сказати, що він повністю гарантує стійке збереження якості звучання на реалізованих швидкостях потоку, або що для нього написано таке ж безліч зручного програмного забезпечення, як для MP3.

### 1.1.3 Free Lossless Audio Codec

Формат FLAC був розроблений співробітниками веб-сайту Xiph.org. Його використовували для стиснення аудіофайлів. Стиснення не призводить до втрати якості,

що означає, що при виконанні цього процесу дані не втрачаються. Формат FLAC дозволяє користувачам зберігати оригінальну якість файлів; саме ця властивість перетворює формат в ідеальний метод резервного копіювання аудіо, тому що фізичні носії звуку з часом можуть зіпсуватися. Файли FLAC використовують для резервного копіювання високоякісного звуку з компакт-диска, тому що якість файлів MP3 залишає бажати кращого. Ліцензія на файли FLAC абсолютно безкоштовна і загальнодоступна. Можливе додавання в формат коштів перевірки цілісності записи, метаданих та зображень.

Завдяки високій швидкості кодування файли FLAC дуже часто мають розмір, який на не менше ніж 50% менше розміру оригінального файлу. Таке стиснення, втім, не призводить до будь-якої втрати якості. Часто файли FLAC використовуються для он-лайн мовлення і он-лайн кодування в реальному часі. При цьому проект FLAC містить наступні елементи: формат мовлення, формат контейнера, довідкову бібліотеку кодеків, а також вхідні плагіни. Формат FLAC підтримує семпли з фіксованою крапкою з діапазонами біт PCM від 4 до 32 біт на семпл і частотою дискретизації до 655 350 Гц (к-сть каналів - від 1 до 8). При додаванні нових полів існуючі декодери не відчувають ніякого впливу завдяки захисту файлів FLAC.

## 1.2 Призначення та область застосування

Завдання дипломного проєкту — розробити продукт для представлення звукових даних у графічному вигляді, їх обробки, та подальшого кодування.

Основним завданням продукту є графічне представлення та редагування WAV файлів.

Програмний продукт повинен виконувати такі задачі:

а) генерація звукових даних;

б) реалізація роботи із файловою системою;

в) реалізація графічного представлення звукових даних;

г) реалізація можливостей звукового програвача;

д) реалізація операцій обробки звукових даних у режимі реального часу:

1) копіювання;

2) вставка;

3) вирізання;

е) реалізація графічного інтерфейсу застосунку;

ж) реалізація кодування у формати WAV, MP3, FLAC;

з) зміна масштабу, перегляд окремих ділянок аудіо;

и) інформування користувача про стан опрацювання команд;

## 1.3 Висновки до розділу

У цьому розділі була досліджена структура різних форматів представлення звукових даних для подальшої можливості роботи з ними. Було виявлено основні особливості кодування: розмір заголовків та шматків даних, особливості кодування та графічного представлення. Також було сформульовано основні задачі, які повинен вирішувати застосунок, створений у ході роботи дипломного проєкта.

# 2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

На початку розробки було досліджено декілька існуючих рішень для обробки звукових даних. Нижче наведено аналіз існуючих рішень для виокремлення їх сильних та слабких сторін.

## 2.1 «AIMP»

«AIMP» — безкоштовний застосунок для програвання та конвертації звукових файлів, зображений на рисунку 2.1. Підтримує безліч форматів звукових файлів (wav, flac, mp3, aac, ape тощо) та можливість кодування у зазначені формати. Цей застосунок також надає можливість обробки звукових даних завдяки вбудованим аудіоефектам, таким як реверб, приглушення, налаштування темпу та дозволяє використати 18-смуговий еквалайзер. AIMP є дуже потужним застосунком завдяки додатковим функціям редактора аудіотегів та таймера вимкнення компьютера. Підтримує сторонні модулі та можливість налаштування користувацького інтерфейсу. До недоліків можна віднести відсутність можливості побітового редагування аудіофайлів та недостатню швидкодію при кодуванні звукових даних.


Рисунок 2.1 — Інтерфейс AIMP

## 2.2 «FL Studio»

FL Studio — комплексна цифрова звукова робоча станція, що розповсюджується на платній основі. FL Studio є найбільш популярним застосунком для запису та зведення звукових доріжок. Цей застосунок надає широкий спектр інструментів для редагування та створення аудіофайлів та підтримує можливість кодування у велику кількість форматів, у тому числі пропрієтарних (.flp). FL Studio постачається із великим набором плагінів, що дозволяють редагувати та генерувати звукові дані у режимі реального часу.

На рисунку 2.2 зображено інтерфейс робочого місця у FL Studio. Це повністю професійний інструмент і він не буде придатний для користувачів, не знайомих із теорією обробки та мікшування звукових доріжок. Варто помітити, що загальна комплексність робить FL Studio дуже вибагливим до ресурсів застосунком та погіршує швидкодію при кодуванні аудіофайлів.

Рисунок 2.2 — Інтерфейс FL Studio

2.3 «foobar2000»

«foobar2000» — безкоштовний аудіпрогравач з широким спектром можливостей конвертаціїї та додаткових налаштувань. Розроблений польським програмістом з Nullsoft, цей застосунок отримав деяку популярність у серед досвідчених користувачів завдяки своїй невибагливості до ресурсів та можливості удосконалення. До беззаперечних переваг застосунку можна віднести можливість прямого виводу даних на звукову карту, підтримку сторонніх програмних модулів та досить високу, порівнянно із іншими застосунками швидкість кодування звукових файлів. Не зважаючи на усі можливості цього програвача, він не позбавлений недоліків, що ускладнює комплексну роботу із звуковими даними. Такими є застарілий інтерфейс користувача, продемонстрований на рисунку 2.3 та відсутність функції редагування звукових файлів у режимі реального часу.

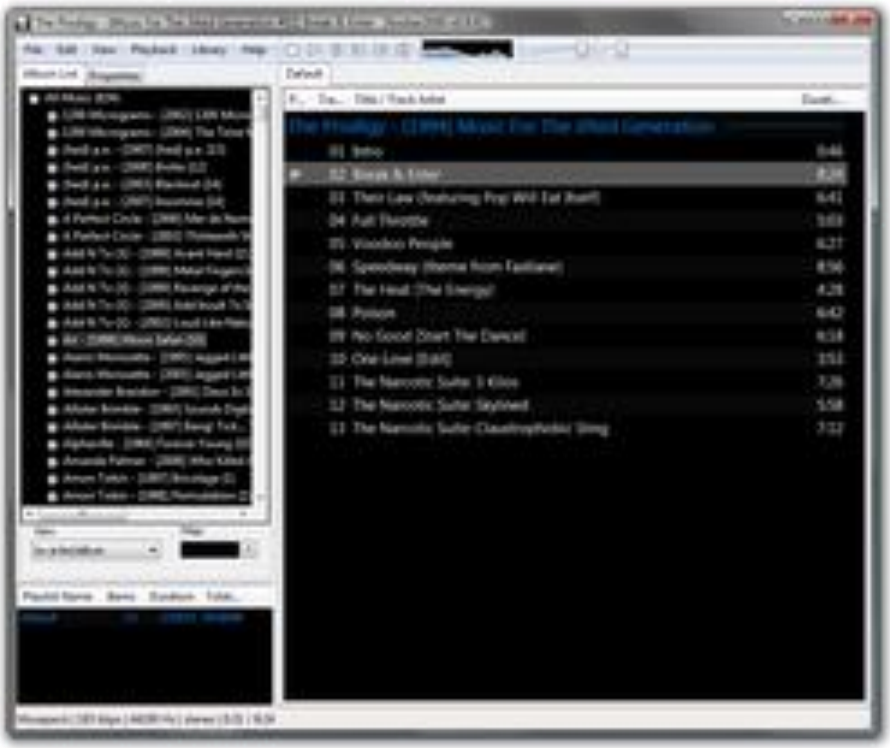

Рисунок 2.3 — Інтерфейс foobar2000

## 2.4 «Winamp»

Один із найбільш популярних застосунків для програвання аудіо, Winamp був створений у 1998 році компанією Nullsoft. Завдяки використанню умовно-безкоштовної моделі розповсюдження та відсутності сильних конкурентів, Winamp зміг швидко набрати досить велику базу користувачів, тим самим забезпечивши можливість подальшого розвитку продукту. Архітектура Winamp дозволяє розширювати функціонал застосунку за допомогою плагінів, а також налаштувати вигляд інтерфейсу користувача. Безперечною перевагою цього застосунку є наявність еквалайзера, підтримка великої кількості форматів аудіо для програвання та конвертації. Із наявних недоліків можна виділити необхідність придбання преміум-версії застосунку для кодування у формат MP3.


Рисунок 2.4 — Інтерфейс Winamp

## 2.5 Висновки до розділу

У цьому розділі було проведено аналіз вже існуючих рішень та виділено їх основні переваги та недоліки. Загалом, функціонал аналогів є цілком обмеженим та розрахованим на певні категорії користувачів. При розробці власного застосунку необхідно звертати увагу на технічні можливості системи та різноматність функцій.

# 3 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Після аналізу предметної області та огляду вже існуючих рішень, було сформовано такі вимоги до функціоналу додатку. Вони наведені у таблиці 3.1.

Таблиця 3.1 — Вимоги до функціоналу додатку

| Номер | Назва |
|---|---|
| UC01 | Генерація звукових даних |
| UC02 | Відкриття аудіофайлу |
| UC03 | Програвання звукових даних |
| UC04 | Призупинка програвання |
| UC05 | Зупинка програвання |
| UC06 | Копіювання фрагменту звукових даних |
| UC07 | Вирізання фрагменту звукових даних |
| UC08 | Вставка звукових даних |
| UC09 | Програвання аудіо із зацикленням |
| UC10 | Регулювання швидкості програвання |
| UC11 | Регулювання гучності програвання |
| UC12 | Кодування звукових даних |
| UC13 | Перегляд окремих ділянок файлу |
| UC14 | Збереження аудіофайлу |

У таблицях 3.2 — 3.15 детально розглядаються варіанти використання додатку.

Таблиця 3.2 — Варіант використання UC01

| | |
|---|---|
| Назва | Генерація звукових даних |
| Опис | Користувач повинен мати можливість згенерувати звуковий файл за заданим шаблоном. |
| Агенти | Користувач |
| Передумови | — |
| Післяумови | Звукові дані успішно згенеровані користувачем. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач натискає кнопку «Згенерувати Аудіо»;<br>3. Система генерує сигнал за шаблоном. |
| Розширений сценарій | — |

Таблиця 3.3 — Варіант використання UC02

| | |
|---|---|
| Назва | Відкриття аудіофайлу |
| Опис | Потрібно надати користувачеві можливість відкрити аудіофайл із файлової системи. |
| Агенти | Користувач |
| Передумови | — |

| | |
|---|---|
| Післяумови | Аудіофайл успішно відкритий користувачем та представлений у вигляді звукової хвилі. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач натискає кнопку «Відкрити Аудіо»;<br>3. Із контекстного меню користувач обирає аудіофайл.<br>4. Система відкриває аудіофайл для подальшої роботи.<br>5. Система зображає аудіофайл у вигляді звукової хвилі. |
| Розширений сценарій | 4.1 Система виявляє, що формат файлу не підтримується;<br>4.2 Система відображає повідомлення про помилку. |

Таблиця 3.4 — Варіант використання UC03

| | |
|---|---|
| Назва | Програвання звукових даних |
| Опис | Користувач повинен мати можливість прослухати аудіофайл. |
| Агенти | Користувач |
| Передумови | Аудіофайл повинен бути згенерований або відкритий користувачем. |

| | |
|---|---|
| Післяумови | Користувач насолоджується звуком аудіо. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач натискає кнопку «Програти Аудіо»;<br>3. Система починає програвання аудіофайлу. |
| Розширений сценарій | — |

Таблиця 3.5 — Варіант використання UC04

| | |
|---|---|
| Назва | Призупинка програвання |
| Опис | Користувач повинен мати можливість призупинити програвання аудіофайлу. |
| Агенти | Користувач |
| Передумови | Користувач повинен почати програвання аудіофайлу. |
| Післяумови | Користувач призупиняє програвання аудіо і може відновити програвання з місця зупинки. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач натискає кнопку «Призупинити Аудіо»; |

| | |
|---|---|
| | 3. Система призупиняє програвання аудіофайлу. |
| Розширений сценарій | — |

Таблиця 3.6 — Варіант використання UC05

| | |
|---|---|
| Назва | Зупинка програвання |
| Опис | Користувач повинен мати можливість зупинити програвання аудіофайлу. |
| Агенти | Користувач |
| Передумови | Користувач повинен почати програвання аудіофайлу. |
| Післяумови | Користувач зупиняє програвання аудіо і може відновити програвання з початку. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач натискає кнопку «Зупинити Аудіо»;<br>3. Система зупиняє програвання аудіофайлу. |
| Розширений сценарій | — |

Таблиця 3.7 — Варіант використання UC06

| | |
|---|---|
| Назва | Копіювання фрагменту звукових данних |

| | |
|---|---|
| Опис | Необхідно надати можливість реалізації операцій копіювання звукових даних у режимі реального часу. |
| Агенти | Користувач |
| Передумови | Користувач повинен відкрити або згенерувати звукові дані. |
| Післяумови | Користувач копіює звукові дані до буфера обміну додатку і може вставити їх у вибране місце. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач виділяє фрагмент даних із елемента управління звукової хвилі;<br>3. Користувач натискає кнопку «Копіювати»;<br>4. Система розміщує фрагмент даних до буфера обміну. |
| Розширений сценарій | — |

Таблиця 3.8 — Варіант використання UC07

| | |
|---|---|
| Назва | Вирізання фрагменту звукових даних. |
| Опис | Користувачеві необхідно надати можливість вирізання звукових даних. |
| Агенти | Користувач |

| | |
|---|---|
| Передумови | Користувач повинен відкрити або згенерувати звукові дані. |
| Післяумови | Користувач вирізає звукові дані із файлу до буфера обміну додатку і може вставити їх у вибране місце. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач виділяє фрагмент даних із елемента управління звукової хвилі;<br>3. Користувач натискає кнопку «Вирізати»;<br>4. Система розміщує фрагмент даних до буфера обміну та видаляє його із файлу. |
| Розширений сценарій | — |

Таблиця 3.9 — Варіант використання UC08

| | |
|---|---|
| Назва | Вставка фрагмента звукових даних |
| Опис | Потрібно реалізувати можливість вставки фрагмента звукових даних. |
| Агенти | Користувач |
| Передумови | Користувач повинен скопіювати або вирізати фрагмент звукових даних. |
| Післяумови | Фрагмент даних вставлено у аудіофайл. |

| | |
|---|---|
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач натискає на обране місце у елементі управління;<br>3. Користувач натискає кнопку «Вставити».<br>3. Система вставляє фрагмент даних у обране місце. |
| Розширений сценарій | 3.1 Система виявляє, що буфер обміну пустий;<br>3.2 Система відображає повідомлення про помилку. |

Таблиця 3.10 — Варіант використання UC09

| | |
|---|---|
| Назва | Програвання аудіо із зацикленням |
| Опис | Необхідно реалізувати можливість зацикленного програвання аудіофайлів. |
| Агенти | Користувач |
| Передумови | Користувач повинен відкрити або згенерувати аудіофайл. |
| Післяумови | Користувач насолоджується програванням аудіо у зацикленному режимі. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку; |

| | |
|---|---|
| | 2. Користувач натискає кнопку «Програвання з зацикленням»;<br>3. Система ппочинає програвання аудіофайлу із зацикленням. |
| Розширений сценарій | — |

Таблиця 3.11 — Варіант використання UC10

| | |
|---|---|
| Назва | Регулювання швидкості програвання |
| Опис | Користувач повинен мати можливість змінити швидкість програвання аудіофайлу. |
| Агенти | Користувач |
| Передумови | Користувач повинен почати програвання аудіофайлу. |
| Післяумови | Швидкість програвання змінюється. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач рухає повзунок регулювання швидкості програвання;<br>3. Система змінює швидкість програвання аудіофайлу. |
| Розширений сценарій | — |

Таблиця 3.12 — Варіант використання UC11

| | |
|---|---|
| Назва | Регулювання гучності програвання |

| | |
|---|---|
| Опис | Користувач повинен мати можливість змінити гучність програвання аудіофайлу. |
| Агенти | Користувач |
| Передумови | Користувач повинен почати програвання аудіофайлу. |
| Післяумови | Змінюється гучність програвання. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач рухає повзунок регулювання гучності програвання;<br>3. Система змінює гучність програвання аудіофайлу. |
| Розширений сценарій | — |

Таблиця 3.13 — Варіант використання UC12

| | |
|---|---|
| Назва | Кодування звукових даних |
| Опис | Необхідно реалізувати можливість кодування аудіопотоку у наведені формати: WAV, MP3. |
| Агенти | Користувач |
| Передумови | Користувач повинен відкрити або згенерувати аудіофайл. |

| | |
|---|---|
| Післяумови | Користувач отримує аудіофайл формату WAV або MP3 для подальшої роботи. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач натискає кнопку «Кодувати Аудіо»;<br>3. Система відкриває вікно вибору параметрів кодування.<br>4. Користувач обирає кодек, частоту дискретизації, бітрейт та режим управління каналами та натискає кнопку «Кодувати».<br>5. Система кодує звукові дані, сповіщує користувача про кінець кодування. |
| Розширений сценарій | — |

Таблиця 3.14 — Варіант використання UC13

| | |
|---|---|
| Назва | Перегляд окремих ділянок аудіофайлу |
| Опис | Користувач повинен мати можливість переглянути ділянку файлу та масштабувати її для комфорту. |
| Агенти | Користувач |

| | |
|---|---|
| Передумови | Користувач повинен відкрити або згенерувати аудіофайл. |
| Післяумови | Користувач отримує масштабоване зображення звукових даних. |
| Успішний сценарій | 1. Користувач переходить до інтерфейсу додатку;<br>2. Користувач використовує колесо миші для масштабування даних, подвійне натискання на праву клавішу для повернення до оригінального масштабу.<br>3. Система масштабує дані. |
| Розширений сценарій | — |

### 3.1 Вимоги до інтерфейсу

Підхід до дизайну інтерфейсів постійно змінюється: внаслідок появи нових засобів та методів поширення інформації виникає проблема коректного сприйняття цієї інформації користувачем. Інтерфейс користувача, у першу чергу, повинен бути простим для взаємодії та не перевантаженим надлишковими елементами. Для усунення можливостей некоректного використання, елементи керування необхідно активувати по мірі виконання операцій.

Для вирішення поставлених задач, необхідно створити багатовіконний інтерфейс, окремий для функцій конвертації аудіофайлів. Окрім цього, найважливішою задачею розробки інтерфейсу являється створення компонента графічного зображення

аудіопотоку, оскільки саме за допомогою цього компонента користувач буде обробляти аудіо дані.

## 3.2 Висновки до розділу

У цьому розділі було сформульовано вимоги до програмного забезпечення, створено сценарії використання застосунку та наведено детальний текстовий опис кожного з сценаріїв, були сформовані вимоги до інтерфейсу користувача. У результаті виявлені наступні функціональні можливості застосунку: простий і зрозумілий інтерфейс, можливість редагування аудіозапису у режимі реального часу, підтримка найбільш поширених форматів, гнучкі системні вимоги. Сформульовані вимоги до функціоналу дозволяють представити, як саме буде працювати застосунок, скільки необхідно реалізувати ролей користувачів, які можливості потрібно залишити, а які — видалити через їх непотрібність.

# 4 ПРОЄКТУВАННЯ ЗАСТОСУНКУ

## 4.1 Діаграма використання

Виходячи із функціональних вимог до додатку, що були сформульовані у минулому розділі, можна виділити одного актора системи, а саме Користувача і розробити діаграму використання для графічного зображення варіантів використання додатку. Спроєктовану систему можна зобразити у вигляді множини прецедентів, що позначають всі можливості використання системи користувачем. Усього можна виділити 14 варіантів використання програми. Діаграма використання зображена на кресленику ІТ61.310БАК.004 Д1.

## 4.2 Діаграма послідовності редагування аудіо

Діаграма послідовності редагування аудіо зображена на кресленику ІТ61.310БАК.004 Д2. На ній зображено процес виклику функцій та передачі даних при редагуванні аудіофайлу. З діаграми видно, що процес починається з відкриття аудіофайлу користувачем. Система відкриває файл та декодує його у формат PCM. Після цього, декодовані дані зображаються графічно за допомогою компонента візуалізації. Графічна хвиля відображається у вікні основного інтерфейсу користувача, після чого користувач отримує доступ до всіх можливостей застосунку: конвертації, програвання та редагування. Редагування аудіо тісно пов’язане із функціоналом компонента візуалізації і починається із виділення фрагменту аудіо. Після виділення користувачем фрагменту даних та натиску кнопки «Копіювати», система копіює фрагмент даних до буфера обміну. За натиском кнопки «Вставити», система додає фрагмент даних до PCM аудіо та надсилає команду про збереження аудіофайлу. Після завершення редагування, система зберігає аудіофайл на жорсткому диску та знову візуалізує аудіофайл на основному інтерфейсі користувача.

## 4.3 Діаграма послідовності конвертації аудіо

Діаграма послідовності конвертації аудіо зображена на кресленику ІТ61.310БАК.004 Д3. На ній зображено процес виклику функцій та передачі даних при транскодуванні аудіофайлу. З діаграми видно, що процес починається з вибору та подальшого відкриття аудіофайлу формату MP3 користувачем. Система зчитує файл за допомогою компонента Source Reader та декодує дані у формат аудіо PCM. Після цього, декодовані дані зображаються графічно за допомогою компонента візуалізації. Графічна хвиля відображається у вікні основного інтерфейсу та надає доступ до всіх можливостей додатку. За натиском кнопки «Конвертувати у WAV», система починає процес кодування у WAV. Процес кодування заключається у створенні системою нового файлу із відповідним розширенням, запису заголовку WAVE у початок цього файлу та подальшого побайтового запису даних PCM аудіо.

## 4.4 Діаграма активності

Діаграма діяльності зображена на кресленику ІТ61.310БАК.004 Д5. На ній вказано узагальнений алгоритм роботи програми, від етапу запуску програми до завершення роботи із аудіофайлом. Робота із програмою починається із запуску застосунку користувачем, перед яким з’являється основний інтерфейс. Для роботи необхідно надати аудіоінформацію, яку можна згенерувати за шаблоном або відкрити із файлу, на вибір коистувача. При виборі файлу для відкриття, допустими форматами є WAV, MP3. Якщо користувач обирає WAV — система просто зчитує аудіо дані, що зберігаються у Data Chunk, одразу після 44 байтів даних заголовку. У випадку вибору файлу формату MP3, система декодує дані у PCM. Після отримання аудіо даних, система зображує їх графічно за допомогою компонента візуалізації та виводить зображення на екран користувачеві для подальшої роботи. Далі, користувач може редагувати аудіо дані, конвертувати їх до вибраного формату та програти отримане аудіо.

Під час програвання аудіо-файлу, користувач може налаштувати параметри, тим самим змінивши швидкість та гучність програвача для отримання необхідного у даний момент звучання.

Процес редагування аудіо заключається у роботі користувача із компонентом візуалізації. На початку, користувач мишкою виділяє фрагмент аудіо. Далі, користувач може скопіювати або вирізати цей фрагмент. Скопійований або вирізаний фрагмент можна вставити у аудіофайл. При вирізанні даних, їх також можна видалити. Після завершення процесу редагування, аудіо дані зберігаються.

Процес конвертації аудіо відрізняється у залежності від формату кінцевого файлу. Оскільки застосунок зберігає дані у нестисненому форматі PCM, для конвертації до WAV необхідно лише записати у початок файлу заголовок та скопіювати туди саме власне дані. Результатом успішної конвертації є створений у файловій системі аудіо-файл формату WAV. Для конвертації даних у формат MP3, система передає дані до компонента Sink Writer, який здійснює кодування у заданий формат. Результатом буде створений файл формату MP3.

Після закінчення роботи з програмою, користувач зберігає дані та закриває застосунок.

## 4.5 Діаграма класів

Діаграма класів застосунку зображена на кресленику ІТ61.310БАК.004 Д4. На ній представлена загальна структура моделі, тобто класи та їх зміст, функції, змінні, типи даних. Діаграма класів служить графічним представленням статичної структури застосунку у об’єктно-орієнтованій парадигмі. Окрім власне візуалізації та документування структури ієрархії класів, діаграма класів також використовується для конструювання шляхом прямого або зворотнього проектування. Розроблена діаграма містить 11 класів, що необхідні для коректної роботи застосунку, та показує відношення

у цих класах. Детальний опис класів, зображених на діаграмі, представлений у наступних розділах.

## 4.6 Висновки до розділу

У цьому розділі описується процес розробки діаграм для подальшого створення застосунку. Під час проєктування застосунку, було розроблено діаграму використання, діаграми послідовностей редагування та конвертації аудіофайлу. Також було створено діаграму класів, опис якої надається у наступних розділах пояснювальної записки. Окрім цього, також було розроблено та описано діаграму активності, що пояснює загальний алгоритм роботи застосунку. Побудова діаграм є важливим етапом розробки, тому що це надає змогу графічно представити, як буде функціонувати система у результаті ще до початку реалізації. Завдяки коректно спроектованим діаграмам, можна виділити фундаментальні елементи системи та чітко розпланувати роботу над створенням застосунку.

# 5 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

## 5.1 Огляд технологій для розробки застосунку

Сучасні технології пропонують широкий вибір інструментів для виконання поставлених цілей. Серед найбільш популярних для розробки клієнтських застосунків мов програмування, можна виділити C#, Java. Функціонал конвертера аудіофайлів залежить від доступних кодеків. Для створення такого функцоналу, доречно використовувати різноманітні API Windows, що дають доступ до вже встановлених на комп'ютері користувача кодеків. До таких API відносяться Audio Compression Manager, DirectX Media Objects та Microsoft Media Foundation.

З огляду на сформовані вимоги до функціоналу, для створення вищезазначеного застосунку було обрано фреймворк .Net, об’єктно-орієнтовану мову програмування C#, технологію Windows Presentation Foundation (WPF) для створення інтерфейсу користувача та компонента графічного представлення звукових даних. Для розробки конвертера аудіо було обрано API Media Foundation.

### 5.1.1 C# та Visual Studio

Середовищем розробки для мови C# було обрано інтегроване середовище розробки (IDE) Visual Studio 2019, створену компанією Microsoft. Інтегроване середовище розробки є многофункціональним програмним забезпеченням, створеним для задоволення потреб розробників щодо можливостей покращення їх додатків. Окрім стандартного редактора коду та дебагера, що існують у більшості сучаних IDE, Visual Studio також містить додаткові інструменти, призначені для підвищення ефективності роботи розробника, такі як компілятор, графічні конструктори діаграм та засоби автозавершення коду. Окрім цього, дуже доцільною є можливість використання сторонніх бібліотек та додаткових модулів, що поширюються за допомогою менеджера пакетів NuGet. Найбільш корисним засобом для написання коду є IntelliSense для Visual

Studio. IntelliSense надає можливості отримання додаткових відомостей щодо написаного коду, автоматичної генерації коду та усунення помилок при написанні коду.

### 5.1.2 Windows Presentation Foundation

Для розробки графічного інтерфейсу було використано платформу Windows Presentation Foundation. WPF є спеціальною платформою для створення клієнтських додатків для персональних комп'ютерів під управлінням ОС Windows. Платформа розробки підтримує великий набір інструментів та компонентів для розробки додатків, що дозволяє розробити привабливий та доступний для користувача інтерфейс. WPF надає гнучкий набір графічних функцій та володіє наступними перевагами:

- незалежність графіки від роздільної здатності та пристрою, що обумовлюється використанням апаратно-незалежних пікселів розміром 1/96 дюйма, кожен із яких автоматично масштабується;
- підвищення точності системи координат завдяки використанню чисел подвійної точності із плаваючою комою замість чисел одинарної точності;
- підтримка розширенної палітри кольорів sRGB та вбудована можливість управління вхідними даними із різних кольорових палітр;
- автоматичне управління анімованими сценами та підтримка альфа-версії компоновки спрощує програмування графіки;
- використання можливостей графічного процесора для зниження навантаження на центральний процесор.

Платформа WPF підтримує мову розмітки XAML (eXtensible Application Markup Language) для створення та налаштування інтерфейсу користувача декларативним шляхом. Використання XAML для розробки клієнтських додатків надає такі переваги:

- можливість відділити графічний інтерфейс від бізнес-логіки, завдяки чому різні спеціалісти можуть працювати над різними частинами додатку без необхідності постійного втручання;

- компактність та легка подальша підтримка проєкту, написаного за допомогою XAML;
- спрощена локалізація додатків.

### 5.1.3 Windows Media Foundation

Windows Media Foundation — це новий зручний фреймворк та комплекс API, створений для роботи із мультимедіа даними, вперше використаний у ОС Windows Vista. На відміну від більш застарілих фреймворків, він надає розробникам набагато більше можливостей та інструментарій для управління потоком даних у застосунках. Основний принцип роботи WMF полягає у ефективній передачі даних шляхом поступового їх надсилання у вигляді послідовності семплів (Media Sample). Семпли не є медіа даними, а лише контейнером для ще одного контейнера — буфера (Media Buffer), який власне зберігає медіа дані. У WMF існує три типи фундаментальних сутностей для обробки даних, що відповідають за створення даних, їх обробку та подальший вивід:

- Media Source є об’єктом, що відповідає за генерацію медіа даних у системі. Дані можуть бути зчитані із файлу системи, мережевого потоку або пристрою (мікрофон, веб-камера). Цей об’єкт реалізує інтерфейс IMFMediaSource.
- Media Foundation Transforms (MFTs) виконує функції обробки медіа даних та реалізує інтерфейс IMFTransform. Кодеки у застосунках WMF імплементуються саме як MFTs.
- Media Sink є об’єктом, що отримують медіа дані та здійснюють їх вивід до пристрою або запис у файл. Реалізує інтерфейс IMFMediaSink.

Окрім роботи з аудіоданими, WMF також підтримує можливості кодування та декодування відеофайлів. На відміну від Audio Compression Manager, цей фреймворк підтримує можливість розробки мобільних додатків на ОС Windows Phone, а також створення мультимедійних додатків для Windows Store. Безперечною перевагою також можна вважати можливість як декодувати, так і кодувати дані у формат MP3, що

недоступно при використанні ACM. Критичним недоліком вказаного фреймворку є неможливість прямого використання фреймворку із мовою C#. Бібліотека MF.NET була розроблена для того, щоб виправити цей недолік та надати можливість використання WMF у застосунках, написаних мовою C#. Вона використовує технологію Interop Services для того, щоб імплементувати всі функції WMF у середовище .NET.

## 5.2 Висновки до розділу

Після аналізу доступних технологій розробки десктопних застосунків для сімейства ОС Windows, було обрано мову C#. Відповідно до цього, у якості фреймворку було обрано .NET, у поєднанні з платформою Windows Presentation Foundation для створення привабливого користувацького інтерфейсу. Для роботи із аудіо даними було обрано платформу Windows Media Foundation, а для полегшення розробки застосунку використано можливості бібліотеки MF.NET, що надає можливість працювати із обраною платформою у середовищі .NET.

# 6 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 6.1 Структура даних форматів аудіофайлів

Для успішної реалізації можливостей роботи із аудіофайлами, необхідно імплементувати їх внутрішню структуру.

WAV-файл використовує стандартну RIFF-структуру, яка групує вміст файлу з окремих секцій (chunks) - формат вибірок аудіо, аудіо дані, тощо. Кожна секція має свій окремий заголовок секції та окремі дані секції. Тема секції вказує на тип секції і кількість байт, що містяться в секції. Такий принцип організації дозволяє програмам аналізувати тільки необхідні секції, пропускаючи інші секції, які не відомі чи які не вимагають обробки. Деякі певні секції можуть мати в своєму складі підсекції (sub-chunks). Наприклад, секції "fmt" і "data" є підсекціями секції "RIFF". Заголовок WAV файлу займає 44 байти, його структура наведена у таблиці 6.1.

Таблиця 6.1 — Структура даних файлу WAV

| Секція | Зсув файлу (байт) | Розмір поля (байт) | Назва поля | Опис |
|---|---|---|---|---|
| | 0 | 4 | ChunkID | Містить літери «RIFF», записані у ASCII. |
| | 4 | 4 | ChunkSize | Розмір усього файлу -8 байт. |
| | 8 | 4 | Format | Заголовок типу файлу «WAVE» |
| | 12 | 4 | SubchunkID | Заголовок підсекції «fmt» |
| | 16 | 4 | Subchunk1Size | Розмір частини підсекції після цього поля. Для ІКМ = 16. |
| | 20 | 2 | AudioFormat | Для ІКМ = 1. |
| | 22 | 2 | NumChannels | Моно = 1, Стерео = 2 |

| Секція | Зсув файлу (байт) | Розмір поля (байт) | Назва поля | Опис |
|---|---|---|---|---|
| fmt | 24 | 4 | SampleRate | Частота дискретизації. Стандартні значення — 8000, 44100(CD), тощо. |
| | 28 | 4 | ByteRate | SampleRate*BitsPerSample*NumChannels/8 |
| | 32 | 2 | BlockAlign | NumChannels*BitsPerSample/8 |
| | 34 | 2 | BitsPerSample | Кількість біт на семпл, стандартне значення — 16 |
| | 36 | 4 | Subchunk2ID | Заголовок підсекції «data» |
| | 40 | 4 | Subchunk2Size | Розмір підсекції «data» |
| | 44 | * | Data | Власне аудіо дані |

Файл формату MP3 складається із потоку послідовно повторюваних “фреймів”, кожен із яких складається із заголовку фрейму та власне стиснених аудіо даних. Фрейми також можуть супроводжуватися CRC-сумою, що має довжину 16 біт та, за наявності, розташована одразу після заголовку фрейму. Заголовок фрейму має довжину в 32 біти (4 байти). Значення перших 12 біт заголовку завжди дорівнює 1 і складає “слово синхронізації”. Структура даних фрейму MP3 наведена у таблиці 6.2.

Таблиця 6.2 — Структура даних фрейму MP3

| Зсув файлу (біт) | Назва поля | Розмір поля (біт) | Опис |
|---|---|---|---|
| 0 | Sync Word | 12 | Значення усіх бітів завжди дорівнює 1. |

| Зсув файлу (біт) | Назва поля | Розмір поля (біт) | Опис |
|---|---|---|---|
| 12 | Version | 1 | Дорівнює 1 для MPEG-1/2 |
| 13 | Layer | 2 | Рівень стиснення. «01» для Layer-3 |
| 15 | Error Protection | 1 | Захист від помилок за допомогою CRC.<br>00 — захисту немає, 01 — захист наявний. |
| 16 | Bit Rate | 4 | Значення бітрейту. |
| 20 | Frequency | 2 | Частота дискретизаціїї. |
| 22 | Padding Bit | 1 | Біт підкладки, використовується для точної відповідності бітрейту. Якщо значення 1, до фрейма додається ще 1 байт. |
| 23 | Private Bit | 1 | Приватний біт. |
| 24 | Mode | 2 | Режим канального кодування. |
| 26 | Mode Extension | 2 | Використовується тільки для об’єднаного стерео. |
| 28 | Copyright | 1 | Вказує, чи легально копіювати аудіо. |
| 29 | Original | 1 | Вказує, чи належить фрейм оригіналу. |
| 30 | Emphasis | 2 | Вказує декодеру, чи потрібно вирівнювати звучання файлу програмними методами. |
| 32 | Data | * | Власне аудіо дані. |

6.2 Опис архітектури системи

Windows Media Foundation надає можливість використання архітектури Reader-Writer для покращення швидкодії та спрощення процесу конвертації аудіо файлів. Основними компонентами такої архітектури є Source Reader та Sink Writer, що інкапсулють функціонал зчитування та запису медіа даних до файлової системи та можуть автоматично змінювати вхідні або вихідні дані в залежності від необхідного формату. Схема роботи застосунку наведена на рисунку 6.1.

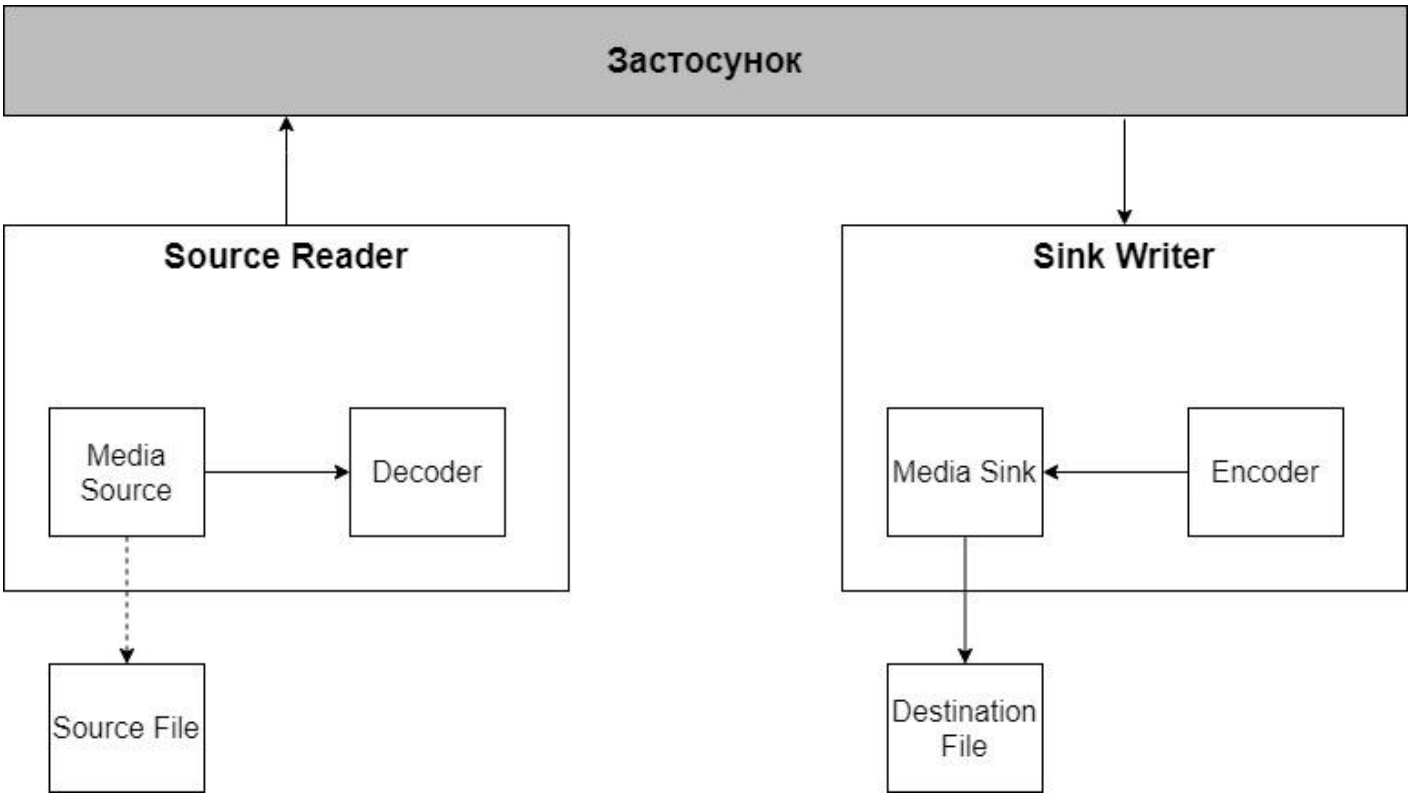


Рисунок 6.1 — Схема роботи застосунку

Нижче наведено більш детальний опис компонентів Source Reader, Sink Writer, включаючи графічні схеми роботи компонентів.

### 6.2.1 Source Reader

Source Reader є імплементацією інтерфейсу IMFSourceReader, призначений для зчитування даних із пристрою або файлу, представлених у вигляді потоку семплів, що містять буфери із декодованими даними. Підтримка декількох потоків дозволяє отримувати із одного файла окремо аудіо та відео дані. Source Reader реалізує 2 структурні об’єкти Media Foundation: Media Source та MFT (кодек). Якщо джерело містить стиснені дані, Source Reader можна використати для декодування. У цьому випадку, компонент самостійно знайде завантажить необхідний кодек та зможе керувати потоком даних між кодеком та джерелом. Source Reader нікуди не надсилає отримані дані, тому далі система повинна прийняти керування і власноруч забирати дані для подальшої обробки. На рисунку 6.2 наведена схема роботи компонента Source Reader.

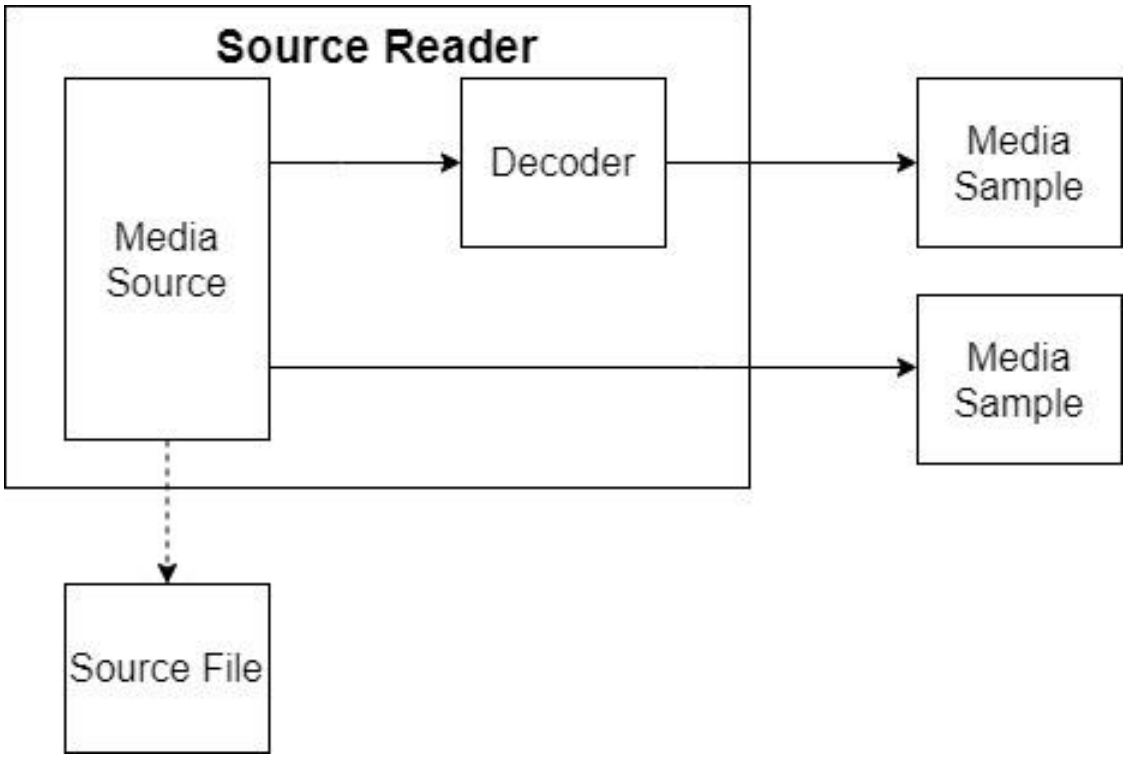


Рисунок 6.2 — Схема роботи Source Reader

### 6.2.2 Sink Writer

Sink Writer являється імплементацією інтерфейсу IMFSinkWriter, та є аналогом компонента Source Reader з єдиною поправкою — замість зчитування даних, цей компонент їх записує. Цей компонент також підтримує можливість кодування даних у стиснені формати для запису у пам’ять комп’ютера. Sink Writer реалізує два структурні компоненти WMF: MFT (кодек) та Media Sink. Часто Sink Writer використовують разом із Source Reader, хоча ці компоненти є взаємонезалежними і можуть використовуватися окремо. На рисунку 6.3 наведена схема роботи компонента Sink Writer.

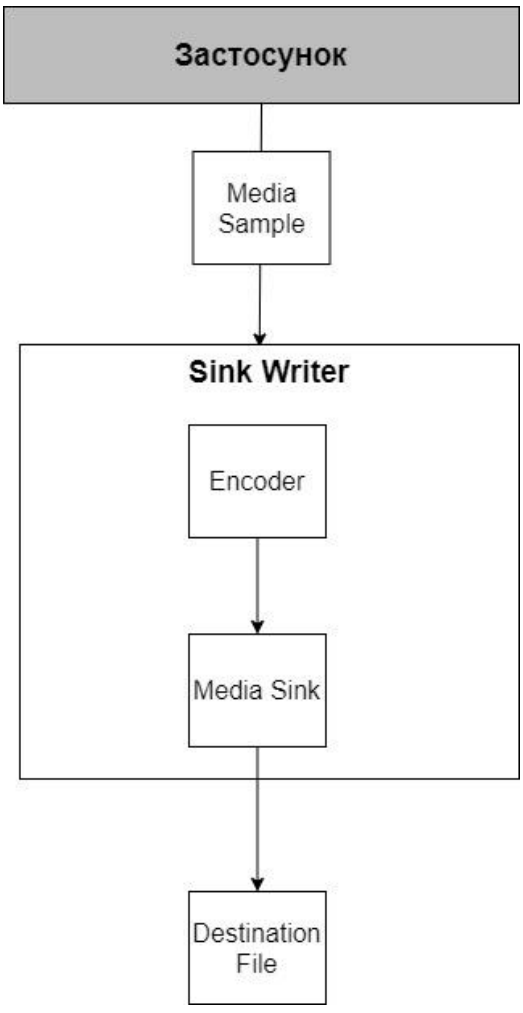


Рисунок 6.3 — Схема роботи Sink Writer

6.3 Опис основних функцій додатку

У ході виконання дипломної роботи було розроблено класи, що описують та реалізують внутрішню структуру звукових файлів, дозволяють кодувати у різні формати, працювати із файловою системою та генерувати звукові коливання з різними параметрами. Функції системи, а точніше обробники подій, що відповідають за управління аудіопотоком на основному інтерфейсі, зображені у таблиці 6.3.

Таблиця 6.3 — Функції класу MainWindow

| Функція | Опис |
|---|---|
| btnGenerateWave_Click | Подає системі команду генерації звукової хвилі за формулою, що вибирається у залежності від значення RadioButton. За вибором користувача, можна згенерувати такі сигнали: синусоїдальний, пилоподібний, квадратну хвилю або трикутну. |
| playBtn_Click | Подає команду програти аудіопотік. |
| pauseBtn_Click | Подає системі команду призупинити програвання аудіопотоку із можливістю продовження подальшого відтворення. |
| openFileBtn_click | За натиском кнопки відкривається діалогове вікно вибору файлу із файлової системи. Далі програвач обирає аудіофайл для відкриття та подальшої роботи. |

| Функція | Опис |
|---|---|
| saveFileBtn_Click | Відкриває діалагове вікно для збереження аудіо до файлової системи користувача |
| copyBtn_Click | Подає команду скопіювати фрагмент даних, виділений на елементі графічного управління. Якщо фрагмент не біло виділено, система видає відповідне повідомлення. |
| cutBtn_Click | Подає команду вирізати фрагмент даних, виділений на елементі графічного управління. Якщо фрагмент не біло виділено, система інформує користувача за допомогою відповідного повідомлення. |
| pastebtn_Click | Подає команду вирізати фрагмент даних, виділений на елементі графічного управління. Якщо фрагмент не біло виділено, система видає відповідне повідомлення. |
| playLoopBtn_Click | Подає системі команду відтворення аудіопотоку у режимі «по колу». У цьому режимі, після досягнення кінця файлу, аудіопоток буде відтворено з початку. |

| Функція | Опис |
|---|---|
| volumeSlider_valueChanged | Подає сигнал для збільшення або зменшення гучності відтворення аудіопотоку. |
| speedSlider_valueChanged | Подає сигнал для збільшення або зменшення швидкості відтворення аудіопотоку |
| convertToMP3btn_Click | Кподає команду для кодування аудіопотоку формату WAV у формат MP3 та відкриває діалогове вікно збереження аудіофайлу. |
| convertToWAVbtn_Click | Подає команду для кодування аудіопотоку формату MP3 у формат WAV та відкриває діалогове вікно збереження аудіофайлу. |

Більш детально опис методів елементу графічного представлення аудіопотоку наведено у таблиці 6.4

Таблиця 6.4 — Опис функцій елемента графічного представлення аудіопотоку

| Назва функції | Опис |
|---|---|
| Read() | Функція для зчитування інформації з аудіофайлу. Якщо файл прочитано успішно, то почнеться обробка інформації та зображення її у графічній формі. |

| Назва функції | Опис |
|---|---|
| WaveControl_Paint() | Функція для зафарбування усього компонента. Заповнює площу компонента фоновим кольором та викликає функцію зображення аудіопотоку. |
| Draw() | Функція зображення аудіопотоку. Визначає кількість семплів файлу на піксель та на основі цього будує масштаб зображення. Для побудови графічного зображення аудіопотоку, алгоритм вираховує максимальне та мінімальне пікове значення амплітуди для вибраного пікселя та проводить лінію між цими точками. |
| Copy() | Функція, що копіює дані з виділеного фрагменту до буферу обміну та дозволяє потім вставити ці дані до потоку. |
| Cut() | Функція, що вирізає дані з виділеного фрагменту до буферу обміну та дозволяє потім вставити ці дані до потоку. |
| Paste() | Функція, що вставляє дані з буферу обміну дані до потоку. |

| Назва функції | Опис |
|---|---|
| ZoomIn() | Ця функція дозволяє збільшити масштаб представлення аудіопотоку шляхом зменшення кількості семплів на один піксель. |
| ZoomOut() | Функція, що дозволяє зменшити масштаб шляхом збільшення кількості семплів на один піксель. |
| ZoomToRegion() | Функція що дозволяє детально переглянути окрему ділянку аудіопотоку шляхом збільшення масштабу у виділеній користувачем зоні. |
| ZoomOutFull() | Функція, що дозволяє повністю зменшити масштаб шляхом збільшення кількості семплів на один піксель та повернутися до перегляду цілого аудіопотоку. |
| Scroll() | Функція, що дозволяє переглядати різні ділянки аудіопотоку у режимі збільшеного масштабу. |

Функції, що відповідають за управління відтворенням аудіопотоку, описані у таблиці 6.5.

Таблиця 6.5 — Опис функцій відтворення аудіопотоку

| Назва | Опис |
|---|---|
| PlayAudio | Відтворює аудіопотік за початку файлу. |
| ResumePlayAudio | Відтворює аудіопотік з місця зупинки. |
| PauseAudio | Призупиняє аудіопотік із можливістю подальшого відтворення у місці зупинки. |
| StopAudio | Повністю зупиняє відтворення аудіопотоку. |
| PlayAudioLoop | Відтворює потік у режимі «по колу», тобто замість завершення відтворення, аудіопотік починає програватися знову. |
| ChangeVolume | Змінює гучність звуку шляхом зміни амплітуди впродовж всього аудіопотоку. |
| SpeedUpX2 | Прискорює відтворення аудіопотоку із коефіцієнтом 2. |
| SpeedDownX2 | Зменшує швидкість відтворення аудіпотоку із коефіцієнтом 2. |

У таблиці 6.6 наведено опис функцій, що відповідають за зчитування файлу формату MP3 із диску та його декодування у PCM, тобто реалізують компонент Source Reader.

Таблиця 6.6 — Реалізація компонента Source Reader для зчитування MP3 файлів

| Назва | Опис |
|---|---|
| MP3FileReader(string filename) | Ця функція є конструктором об’єкту класу MP3FileReader. Під час виклику, ініціює запуск Media Foundation API та викликає функцію створення Source Reader. |
| CreateReader() | Функція, що створює об’єкт класу Source Reader та зчитує дані із файлу на диску. Під час виклику цієї функціїї відбувається налаштування Source Reader, вказується кількість потоків зчитування даних (1 для аудіо) та характеристики вхідного та вихідного типів даних. Повертає об’єкт класу IMFSourceReader. |
| Read(byte[] buffer, int offset, int count) | Перевизначає метод Read() класу Stream. Ця функція дозволяє власне зчитувати дані із файлу. Для цього, спочатку зчитується аудіо семпл, із цього семплу виділяються аудіо буфери, які потом відкриваються для копіювання даних PCM аудіо до вихідного потоку. Це відбувається допоки не буде знайдено кінець аудіо потоку. |

| Назва | Опис |
|---|---|
| ReadFromDecoderBuffer(byte[] buffer, int offset, int needed) | Функція, що повертає кількість байт даних, зчитаних із буфера кодека за один цикл. Використовується для зміни позиції зчитування потоку |

Таблиця 6.7 — Реалізація компонента Sink Writer.

| Назва | Опис |
|---|---|
| EncodeToMP3(Stream stream, string filename, int bitrate) | Ця функція дає системі сигнал підготовки до початку кодування. Під час виклику, відбувається створення компоненту Sink Writer та його налаштування для кодування у формат MP3. Після цього, здійснюється виклик функції PerfromEncode(). |
| CreateSinkWriter(string) | Функція, що створює об’єкт класу IMFSinkWriter та конфігурує його згідно формату вихідного файлу. |
| PerformEncode(IMFSinkWriter writer, int index, Stream stream) | Ця функція здійснює кодування даних із файлу. Для цього, спочатку створюються об’єкти класу IMFSample та IMFMediaBuffer. Дані копіюються до буфера, а потім передаються у семплі до кодека, який записує їх у файл. Це відбувається допоки не буде знайдено кінець аудіо потоку. |

Таблиця 6.8 — Функції для зчитування та запису файлів формату WAV

| Назва | Опис |
|---|---|
| WaveFileWriter(Filename) | Конструктор класу WaveFileWriter. При ініціалізації, створює об’єкт класу Binary Writer та записує до вихідного файлу заголовок WAVE |
| CreateWaveFile(string, Stream):void | Ця функція створює об’єкт класу WaveFileWriter, починає зчитування даних із вхідного потоку до буфера та викликає функцію Write(). |
| Write(byte[], int, int) | Функція, що записує байти даних із буфера до вихідного файлу. Запис відбувається до секції data, що була створена раніше. |
| ReadWaveHeader(Stream): void | Ця функція зчитує дані заголовку файлу WAV та перевіряє наявність всіх належних секцій. Для цього створюється об’єкт класу Binary Reader. |
| ReadDataChunk(Binary Reader): byte[] | Функція, що зчитує власне аудіо дані із секції data та поміщає їх у масив байтів для подальшого використання компонентом візуалізації. |
| GetRiffChunk(Stream, ChunkID, chunkLength): RiffChunk | Функція, що створює та повертає новий об’єкт класу RiffChunk для подальшого зберігання. |

## 6.4 Опис алгоритму конвертації аудіофайлів

Процес конвертації аудіофайлів називається «транскодування» та складається із декількох етапів. Першим етапом є декодування аудіо, тобто перетворення аудіо із початкового формату до PCM. Другим етапом є кодування, тобто перетворення із PCM у потрібний формат.

Спочатку, необхідно створити компонент Source Reader за допомогою функції MP3FileReader(string), яка приймає строку зі шляхом до файлу і надає реалізацію інтерфейсу IMFSourceReader. Перед використанням, Source Reader необхідно налаштувати. тобто вказати потік аудіо даних для обробки, вихідний формат та його характеристики. Source Reader автоматично шукає у системі необхідний кодек в залежності від заданих налаштувань та доступних у системі кодеків. Кодек послідовно декодує дані із файлу та представляє їх у вигляді семплів. Для подальшої конвертації необхідно послідовно зчитувати дані із семплів та записувати байтові дані цих семплів до єдиного буфера. У результаті отримаємо потік нестисненого PCM аудіо.

Далі, для кодування аудіо у формат WAV, необхідно викликати функцію CreateWaveFile(String output, reader), яка приймає строку з іменем кінцевого файлу та дані буфера. При цьому відбувається ініціалізація об’єкту класу BinaryWriter, який записує до вихідного файлу заголовок WAVE, тобто заповнює секції RIFF, fmt та заголовок секції data. Після цього у кінець файлу послідовно записуються байти даних PCM, що були отримані під час декодування. У результаті, отримаємо на диску аудіофайл формату WAV, який можна використовувати для подальшої роботи.

Для зворотньої конвертації, тобто перетворення аудіо із формату WAV у формат MP3, використовується компонент Sink Writer. Спочатку викликається функція EncodeToMP3(stream reader, string filename, int bitrate), яка приймає потік даних, строку з іменем кінцевого файлу та вихідний бітрейт. Ініціюється створення об’єкту Sink Writer через вкладену функцію MFCreateSinkWriterFromURL(string filename), параметри кодування задаються автоматично, в залежності від імені файлу. Після створення Sink

Writer, необхідно налаштувати вхідний та вихідний типи медіа даних, які є об’єктами IMFMediaType.

Процедура кодування достатньо проста: потрібно створити семпл, наповнити його даними із вхідного файлу та передати семпл до функції WriteSample(). Складність операції полягає у тому, що кожен IMFSample містить хоча б один буфер IMFMediaBuffer. Для запису даних у такий буфер, необхідно створити його за допомогою функції MFCreateMediaBuffer(int size), вказавши розмір буфера у байтах. Для усунення помилок кодування блоків протяжністю в 1 секунду, розмір буфера повинен співпадати із значенням AvgBytesPerSecond вхідного файлу. Далі, ми викликаємо функцію Lock() на об’єкт IMFMediaBuffer для того, щоб відкрити доступ до пам’яті буфера та скопіювати фрагмент даних вхідного файлу, після чого викликаємо функцію Unlock() та фіксуємо кількість записаних байт даних. Зрештою, ми викликаємо функцію запису семплу. Цикл продовжується, допоки кількість записаних байт даних є більшою за нуль.

## 6.5 Опис компонента візуалізації аудіопотоку

Аудіопотік може бути зображено графічно, у формі хвилі (waveform), або спектрограми. Форма хвилі є графічним зображенням амплітуди аудіофайлу на протязі часу. Для зображення аудіопотоку у формі хвилі необхідно декодувати сигнал, розбити його на семпли та віднайти значення амплітуд на кожному декодованому семплі. Ці значення розташовуються у якості точок на графіку та з’єднуються для створення графічної форми хвилі. Компонент графічного зображення аудіопотоку було розроблено за допомогою платформи WPF. Його зображено на рисунку 6.4. Окрім власне зображення аудіопотоку, цей компонент дозволяє також:

– змінити масштаб представлення аудіопотоку для більш детального перегляду;

– виділити фрагмент аудіопотоку для реалізації операцій копіювання, вирізання та вставки аудіоданих;

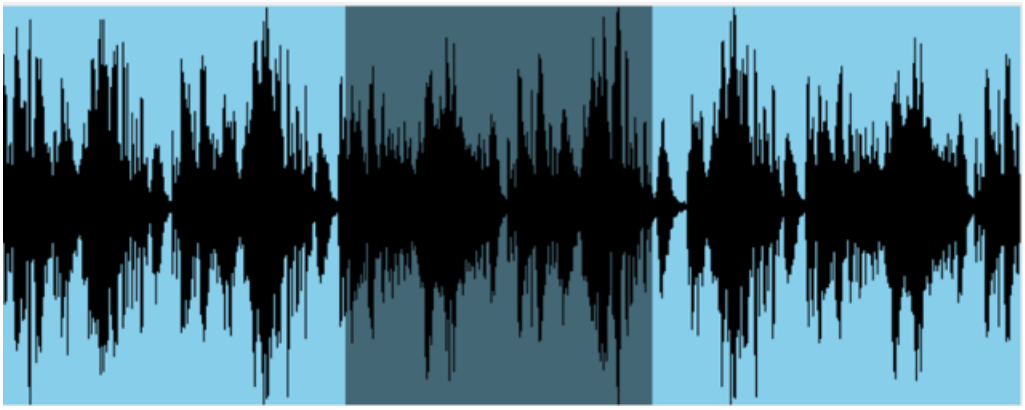

Рисунок 6.4 — Елемент графічного представлення аудіопотоку

## 6.6 Висновки до розділу

У цьому розділі було розглянуто архітектуру системи та наведено схеми роботи окремих її компонентів, таких як Sink Writer, Source Reader. Було виконано детальний опис алгоритму конвертації аудіо даних, тобто перетворення із формату WAV у MP3 та навпаки. Також були описані основні функції системи, включаючи функції відтворення та управління аудіопотоком, функції зчитування та запису даних. Окрім того, у цьому розділі наведено опис компонента графічного представлення аудіопотоку та принцип його роботи. Також було описано імплементовану у систему структуру даних аудіофайлів форматів WAV та MP3.

# 7 ВИКОРИСТАННЯ ЗАСТОСУНКУ

## 7.1 Технічні вимоги до системи

Застосунок для конвертації та програвання аудіофайлів розроблено на платформі .NET із використанням можливостей фреймворку Windows Media Foundation. Відповідно до цього, мінімальні та оптимальні технічні вимоги до системи наведені у таблицях 7.1 та 7.2.

Таблиця 7.1 — Мінімальні технічні вимоги до системи

| | |
|---|---|
| Процесор | 2-ядерний процесор з тактовою частотою більш ніж 1.8 ГГц |
| Жорсткий диск | HDD із 500 МБ вільного простору |
| Оперативна пам’ять | 2 ГБ |
| Аудіокарта | DirectX-сумісна |
| Операційна система | 32-розрядна, Windows 8 та вище |

Таблиця 7.2 — Оптимальні технічні вимоги для системи

| | |
|---|---|
| Процесор | 2-ядерний процесор з тактовою частотою більш ніж 2.1 ГГц |
| Жорсткий диск | SSD із 1 ГБ вільного простору |
| Оперативна пам’ять | 4 ГБ |
| Аудіокарта | DirectX-сумісна |
| Операційна система | 64-розрядна, Windows 8 та вище |

Для стабільної роботи застосунку необхідно забезпечити наявність кодеків для WMF у системі. Це спричиняє неможливість використання застосунку на комп’ютерах

із ОС Windows 7 або більш старими версіями, оскільки можливість кодування аудіо у формат MP3 була реалізована лише у Windows 8. Конвертація аудіофайлів також є вибагливим до потужностей процесом, що накладає певні обмеження на використання застосунку із малопотужними пристроями.

## 7.2 Висновки до розділу

Так як застосунок розроблено для роботи із аудіофайлами, обов’язковою вимогою для системи є наявність аудіокарти, інакше частина функціоналу системи буде недоступна. Із огляду на це, а також на використані технології, було сформульовано мінімальні та оптимальні технічні вимоги до системи. У якості операційної системи можна використовувати лише сімейство Windows. Найбільш оптимальним вибором буде операційна система Windows 10, але не серверний варіант, так як у ньому відсутня підтримка Media Foundation.

# ВИСНОВКИ

З огляду на стан сучасних розробок у сфері кодування аудіо та вивчивши галузь застосування, темою дипломного проекту було обрано створення застосунку для конвертації та програвання аудіофайлів.

Тема дипломного проєкту є актуальною для сьогодення: не зважаючи на популярність новітніх форматів аудіо, що використовуються компаніями для передачі аудіо через Інтернет, MP3 залишається домінуючим форматом кодуванння аудіо для домашнього використання. Основною метою розробки є прагнення зменшити поріг навичок для роботи із аудіофайлами та задовольнити потреби середнього користувача без необхідности використання професійного програмного забезпечення.

Для створення застосунку був проведений ретельний аналіз області та вже існуючих рішень, серед яких: «AIMP» — безкоштовний програвач із функціями конвертера аудіо та редактора аудіотегів, «FL Studio» — повноцінна цифрова робоча станція для професіоналів, та «foobar2000» — безкоштовний застосунок для програвання та конвертації аудіофайлів.

Проведений аналіз дав змогу виокремити основні функціональні вимоги до застосунку та створити на їх основі діаграму варіантів використання, що наочно показує можливості користувача щодо використання програмного забезпечення. Окрім цього, для вдалого проєктування застосунку також було проведено ретельний аналіз, результатом якого стала розробка діаграми класів, діаграми послідовності роботи системи при редагуванні та конвертації аудіофайлу, а також діаграми активності, що відображає загальний алгоритм роботи застосунку від створення аудіо до збереження кінцевого файлу.

Для успішної розробки програмного забезпечення було оглянуто різноманітні технології та вибрано найбільш відповідні до задачі. Виходячи із вимог та доступних технологій, для створення графічної частини системи, а саме інтерфейсу користувача і компонента графічного представлення аудіофайлу, було обрано Windows Presentation

Foundation. Реалізація транскодування аудіо у середовищі .Net стало найбільш важкою задачею, для якої було обрано платформу Windows Media Foundation у поєднанні із бібліотекою MF.NET.

У ході реалізації програмного забезпечення було розроблено застосунок, що використовує архітектуру Reader-Writer для забезпечення надійної та швидкої процедури конвертації аудіофайлів. Окрім цього, було створено компонент для візуалізації аудіофайлів та надано детальний опис системи. Останнім пунктом стало складення детальних технічних вимог до системи.

Результатом виконання проєкту стало створення застосунку, що є оптимальним рішенням для виконання поставленної задачі, оскільки вимагає мінімум ресурсів та реалізує базові можливості роботи із аудіофайлами, такі як можливість графічного зображення звукових даних, їх редагування та конвертація у режимі реального часу та генерація звукових даних за шаблонами. Враховуючи простоту інтерфейсу, можливості користувацького компоненту для відображення та невибагливість до апаратної складової, даний застосунок має перспективи для подальшого розвитку та може зацікавити користувачів. Є можливість покращення додатку, можна збільшити кількість інструментів для редагування та вдосконалити роботу графічного компонента.

## ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Шилдт Г. C# 4.0 Полное Руководство. Москва: ООО «И.Д.Вильямс», 2011. 1056 с.
2. The WAVE File Format URL: http://www.lightlink.com/tjweber/StripWav/WAVE.html (дата звернення 10.05.2020)
3. WAVE PCM soundfile format URL: http://soundfile.sapp.org/doc/WaveFormat/ (дата звернення 07.05.2020)
4. Рисуем .wav-волну. URL: https://habr.com/post/113239/ (дата звернення: 10.05.2020)
5. Радзишевский А. Основи аналогового та цифрового звуку. Москва: «И.Д.Вильямс», 2008. 254 с.
6. John Watkinson. Introduction to Digital Audio. Taylor & Francis, 2002. 419 с.
7. John Watkinson. Art of Digital Audio. Focal Press, 2011. 768 с.
8. Ken Pohlmann. Principles of Digital Audio. McGraw-Hill. 2010. 816 с.
9. Nika Aldrich. Digital Audio Explained: For the Audio Engineer. Sweetwater Sound Inc, 2004. 403 с.
10. Nic Cyn. Windows Media Foundations: Getting Started in C#: A Step-by-Step Guide to Writing Windows Media Foundation Applications in C#. Independently published, 2019. 366 с.
11. Anton Polinger. Developing Microsoft Media Foundation Applications (Developer Reference). Microsoft Press, 2011. 384 с.
12. Julius O. Smith III. Mathematics of the Discrete Fourier Transform (DFT): with Audio Applications. W3K Publishing, 2007. 322 с.
13. Adam Nathan. Windows Presentation Foundation 4 Unleashed. Pearson Education, 2010. 576 с.
14. Matthew MacDonald. Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5. Apress, 2012. 1078 с.

15. AIMP. URL: https://www.aimp.ru/ (дата звернення: 03.05.2020)

16. FL Studio. URL: https://www.image-line.com/flstudio/ (дата звернення: 03.05.2020)

17. foobar2000. URL: https://www.foobar2000.org/ (дата звернення: 03.05.2020)

18. Роберт Сесил Мартин. Чистый код: создание, анализ, рефакторинг. Библиотека программиста, 2003. 464 с.

19. Джон Скит. C# для профессионалов. Тонкости программирования. Вильямс, 2017. 608 с.

20. Bart J.F. De Smet. C# 5 Unleashed. Sams Publishing, 2013. 1728 с.

21. Албахари Бен, Албахари Джозеф. C# 7.0. Справочник. Полное описание языка. Вильямс, 208. 1024 с.

22. WPF Tutorial. URL: https://www.tutorialspoint.com/wpf/index.htm (дата звернення: 04.05.2020)

23. Digital Audio Fundamentals. URL: https://www.pluralsight.com/courses/digital-audio-fundamentals (дата звернення: 04.05.2020)

24. Media Foundation .NET. URL: http://mfnet.sourceforge.net/ (дата звернення: 05.05.2020)